MySQL Performance Tuning für Entwickler

Linux-Tage 2015, Chemnitz

Oli Sennhauser

Senior MySQL Consultant, FromDual GmbH

oli.sennhauser@fromdual.com

FromDual GmbH

www.fromdual.com

Support

Beratung

remote-DBA

Schulung

Datenbank Performance

Über was reden wir eigentlich genau?

- Durchsatz (throughput)
  - z. B. Business-Transaktionen pro Minute
- Antwortzeit (Latenz, response time)
  - z.B. Business-Transaktion dauert 7.2 Sekunden im Schnitt

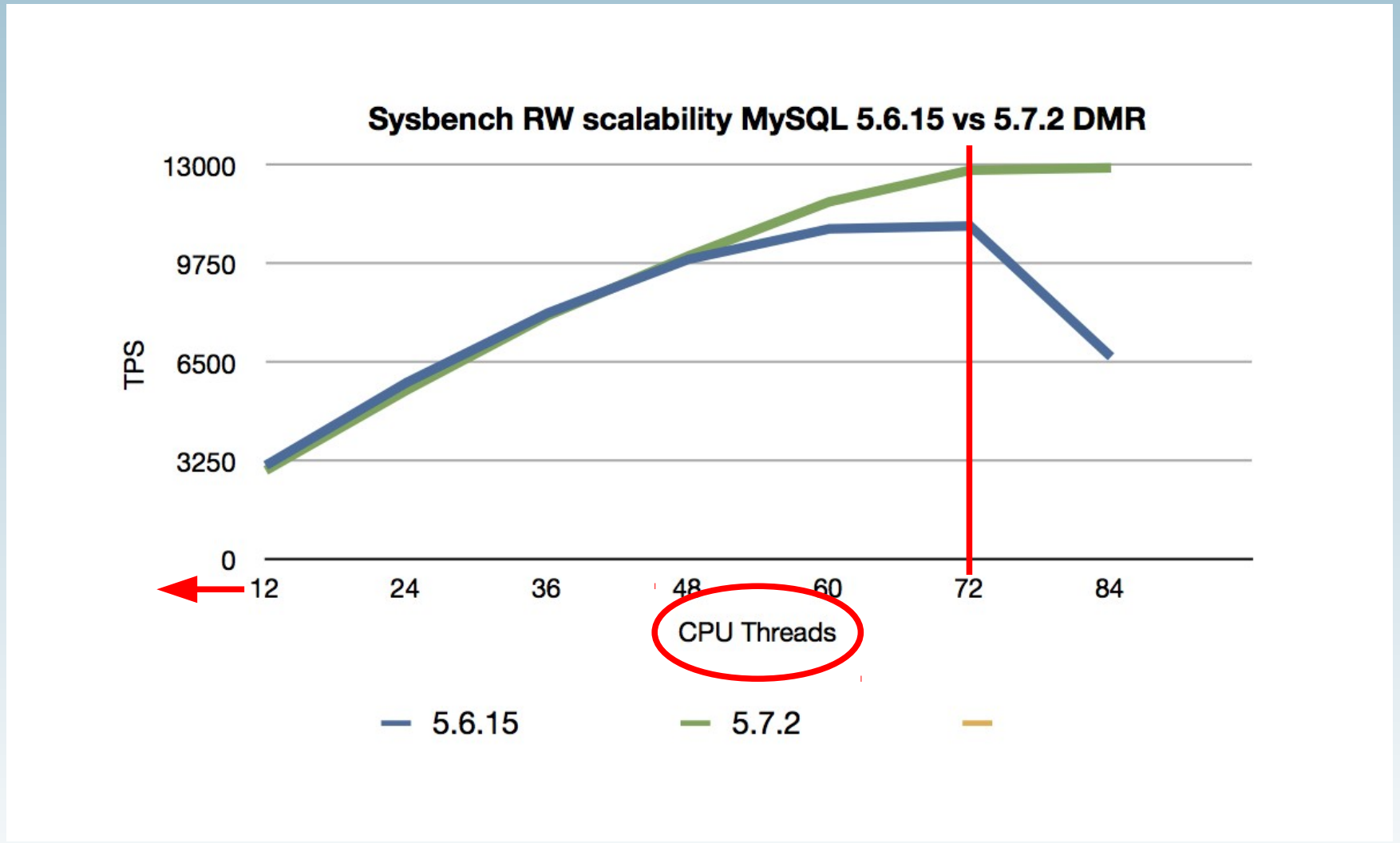
Über was redet Marketing?

- Durchsatz, Skalierbarkeit von DB-Queries

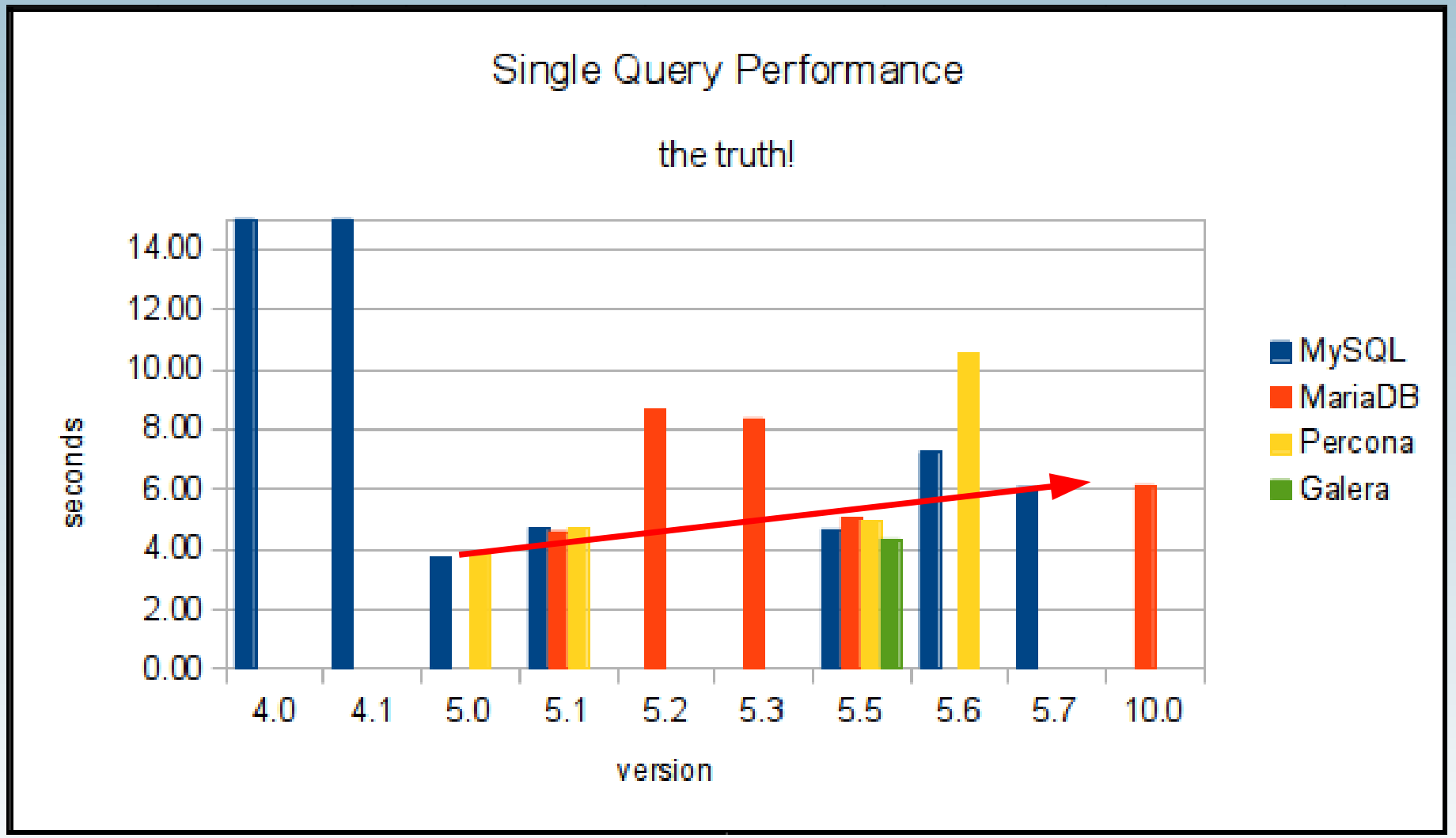
Gap!

- 95% der Nutzer haben ein Latenz-Problem
- 5% ein Durchsatz/Skalierungs-Problem

Durchsatz nimmt zu



Antwortzeit nimmt zu!



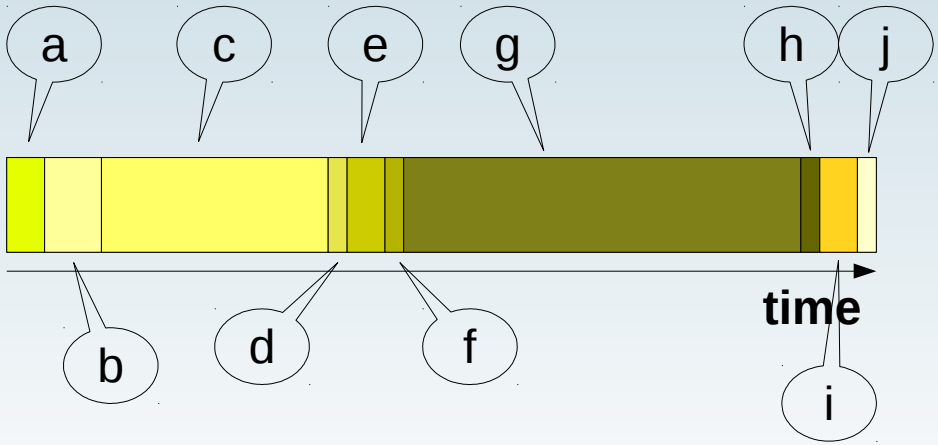
Wo ist meine Zeit geblieben?

www.fromdual.com

- Antwortzeit meiner Business-Transaktion
  - d. h. Zeit messen!!!
  - Applikation mit „Probes“ versehen
  - Profiler (PHP (XDebug), Java (Jprofiler), ...)
  - Profil erstellen:

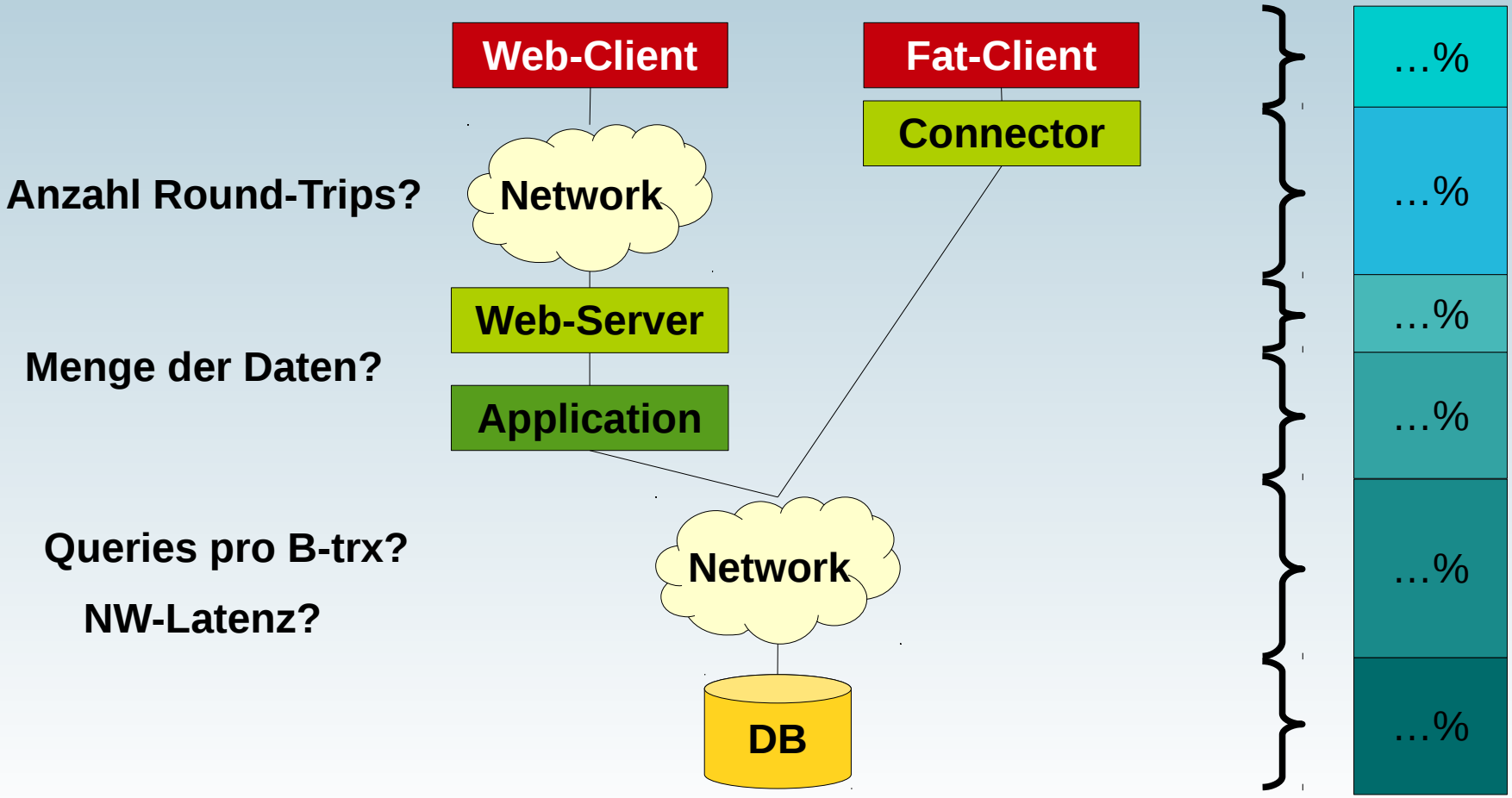
```
function x() {
  start = current_time();
  count[x]++;
  …
  end = current_time();
  duration[x] += (end - start);
}
```

function	count	time	
x	123	156.25	0.8%
y	19	827.30	4.1%
z	2	19280.00	95.1%
Total	144	20263.55	100.0%



End-to-End Profile

- **Idealfall: End-to-End Profile:**
- **Round-Trip pro Business-Transaktion**



General Query Log

- Alle Queries werden gelogged:

```
+------------------+-------------+
| Variable_name    | Value       |
+------------------+-------------+
| general_log      | OFF         |
| general_log_file | general.log |
+------------------+-------------+
```

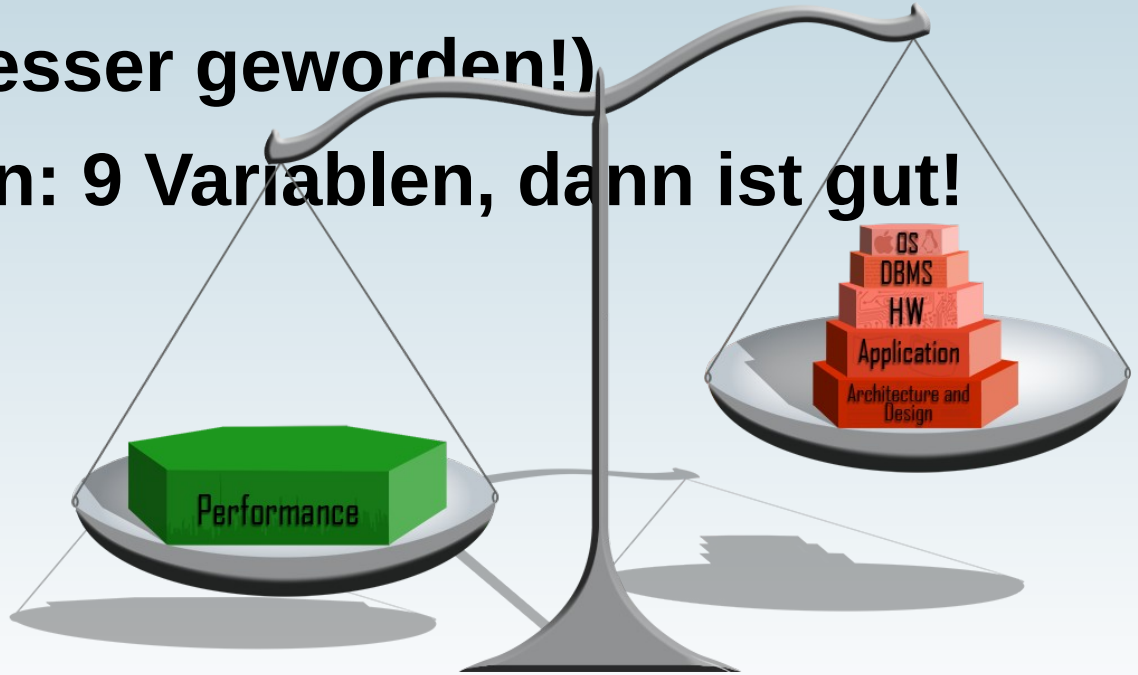
- Gut bei:
  - Frameworks
  - Fremdapplikationen
- Beispiel:
  - CMS: 1 Änderung (30 s)
  - → 30'000 Queries in der DB (ca. 1 ms/Query)

Ist die Datenbank schuld?

- Angenommen die Business-Trx verbringt viel Zeit in der DB:
  - Dann ist NICHT zwingend die DB schuld!
  - SISO Prinzip?
- 1 Connection = 1 Query = 1 Thread = 1 Core
- Heute: Viel Memory, SSD
  → Oft ist/wird wieder die CPU der Flaschenhals
- Wie gucken?
  `vmstat`, `top`, `iostat`

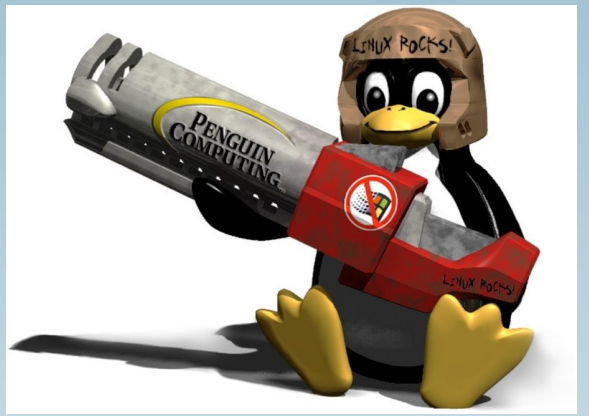
Performance-Waage

- Wo ansetzen?
  - HW, OS, DB Konfiguration, Applikation Architektur und Design
- Typischerweise NICHT DB-Konfiguration
  - (Defaults sind besser geworden!)
  - DB Konfiguration: 9 Variablen, dann ist gut!



Des Admins Bazooka

- Wenig Reaktionszeit:
  - `SHOW [FULL] PROCESSLIST;`
- System entspannen:
  - `KILL [CONNECTION | QUERY] id;`




```
mysql> SHOW PROCESSLIST;
+-----+-------+------+---------+------+-------------+-----------------------------------------------+
| Id  | User  | db   | Command | Time | State       | Info                                          |
+-----+-------+------+---------+------+-------------+-----------------------------------------------+
| 146 | live  | live | Query   |  710 | Sending data | SELECT COUNT(*) FROM (SELECT DISTINCT(nid), ... |
| 240 | live  | live | Query   |  467 | Sending data | SELECT COUNT(*) FROM (SELECT DISTINCT(nid), ... |
| 272 | live  | live | Query   |  275 | Sending data | SELECT COUNT(*) FROM (SELECT DISTINCT(nid), ... |
| 323 | live  | live | Query   |   79 | Sending data | SELECT COUNT(*) FROM (SELECT DISTINCT(nid), ... |
| 374 | admin | NULL | Query   |    0 | NULL         | SHOW PROCESSLIST                              |
+-----+-------+------+---------+------+-------------+-----------------------------------------------+

mysql> KILL CONNECTION 146
```

Slow Query Log

- Etwas systematischer:
  - Slow Query Log

```
+---------------------------------------+----------+
| Variable_name                         | Value    |
+---------------------------------------+----------+
| log_queries_not_using_indexes         | ON       |
| long_query_time                       | 0.500000 |
| slow_query_log                        | OFF      |
| slow_query_log_file                   | slow.log |
| min_examined_row_limit                | 100      |
+---------------------------------------+----------+
```

- Auswerten:
- `mysqldumpslow -s t slow.log > profile`
- `pt-query-digest` (Percona Toolkit)

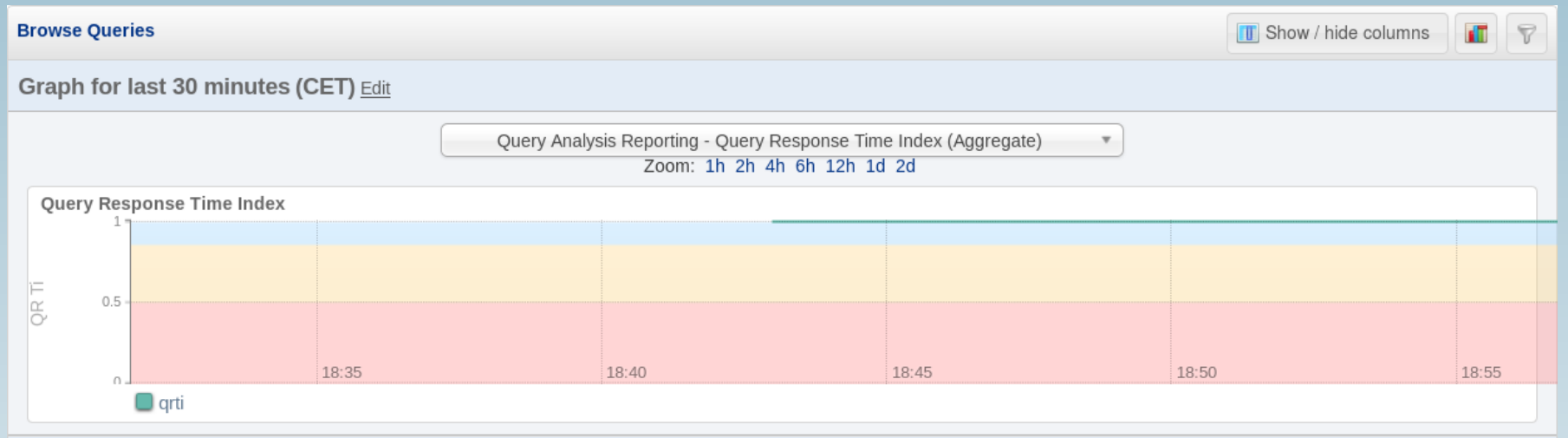
Slow Query Log Profile


```
Count: 4413  Time=2.02s (8902s)  Lock=3.48s (15358s)  Rows=0.0 (0)
, fromdual[fromdual]@2hosts
  UPDATE `accommodationSearch`.`availabilityQueue` SET done = now()
  WHERE accommodationId = N
  AND arrivalDate = 'S' AND duration = N AND availability = 'S'

Count: 124  Time=48.19s (5975s)  Lock=0.01s (1s)  Rows=97.2 (12054)
, fromdual[fromdual]@2hosts
  SELECT ...
    FROM objectdata_view_rucr_history_property_period o
    INNER JOIN ...
   WHERE (o2.value <> N AND o3.begindate <= IF(o.enddate IS NULL OR o.enddate > 'S', 'S', o.enddate)
     AND (o3.enddate IS NULL OR o3.enddate > IF(o.begindate < 'S', 'S', o.begindate))
     AND o.objprop_id = 'S' AND (o.begindate <= 'S' AND (o.enddate IS NULL OR o.enddate >= 'S'))
     AND o2.begindate <= IF(o.enddate IS NULL, 'S', IF(o.enddate > 'S', 'S', o.enddate))
     AND (o2.enddate IS NULL OR o2.enddate > IF(o.begindate < 'S', 'S', o.begindate)))
   HAVING o2.begindate <= o3__1
      AND (o2.enddate IS NULL OR o2.enddate >= o3__0)
    ORDER BY o.begindate ASC, a.accountnumber ASC
```

Graphisch: Query Analyzer

www.fromdual.com



Show 10 entries Export data options...

Showing 1 to 10 of 71 entries First Previous 1 2 3 4 5 Next Last

Query		Counts			QRTi	Latency (hh:mm:ss.ms)				Rows	
		Exec	Err	Warn		Total	Max	Avg	Avg History	Total	Avg
SET SESSION `sql_mode` = ? (1)		1,262	0	0	1.00	0.246	0.004	0.000		0	0
SET SESSION `sql_mode` = ? (1)		1,262	0	0	1.00	0.242	0.004	0.000		0	0
SELECT @@SESSION . sql_mode (1)		1,253	0	0	1.00	0.196	0.003	0.000		1,253	1
SELECT @ @ SESSION . sql_mode (1)		1,253	0	0	1.00	0.191	0.003	0.000		1,253	1
SELECT CAST (SUM_NUMBE...EVENT_NAME = ? LIMIT ? (1)		105	0	0	1.00	0.168	0.009	0.002		105	1
SELECT CAST (SUM_NUMBE...EVENT_NAME = ? LIMIT ? (1)		105	0	0	1.00	0.143	0.010	0.001		105	1
SET autocommit = ? (1)		1,271	0	0	1.00	0.130	0.006	0.000		0	0
SET autocommit = ? (1)		1,271	0	0	1.00	0.127	0.003	0.000		0	0
SET character_set_results = NULL (1)		1,262	0	0	1.00	0.125	0.003	0.000		0	0
SET character_set_results = NULL (1)		1,262	0	0	1.00	0.124	0.003	0.000		0	0

Harte Arbeit

- Sammeln und Schauen (Slow Query Log)
- Verstehen (Query Execution Plan (QEP))
  - `EXPLAIN SELECT COUNT(*) FROM ...`
- Denken
  - Wo lege ich den Index an...

- Tipp 5.7: `EXPLAIN` anderer Connection:

```
EXPLAIN FOR CONNECTION connection_id;
```

Query Execution Plan (QEP)


```
EXPLAIN
SELECT domain
  FROM newsite_domain AS nd
  JOIN newsite_main AS nm ON nd.id = nm.id
 WHERE nm.gbot_indexer = '62'
   AND (nm.state=2 OR nm.state=3 OR nm.state=9)
;
+-------+--------+--------------------+------------+--------------+--------+-------------+
| table | type   | possible_keys      | key        | ref          | rows   | Extra       |
+-------+--------+--------------------+------------+--------------+--------+-------------+
| nm    | range  | PRIMARY,site_state | site_state | NULL         | 150298 | Using where |
| nd    | eq_ref | PRIMARY            | PRIMARY    | jobads.nm.id |      1 |             |
+-------+--------+--------------------+------------+--------------+--------+-------------+

CREATE TABLE `newsite_main` (
  ...
  PRIMARY KEY  (`id`),
  KEY `site_state` (`state`)
);
```

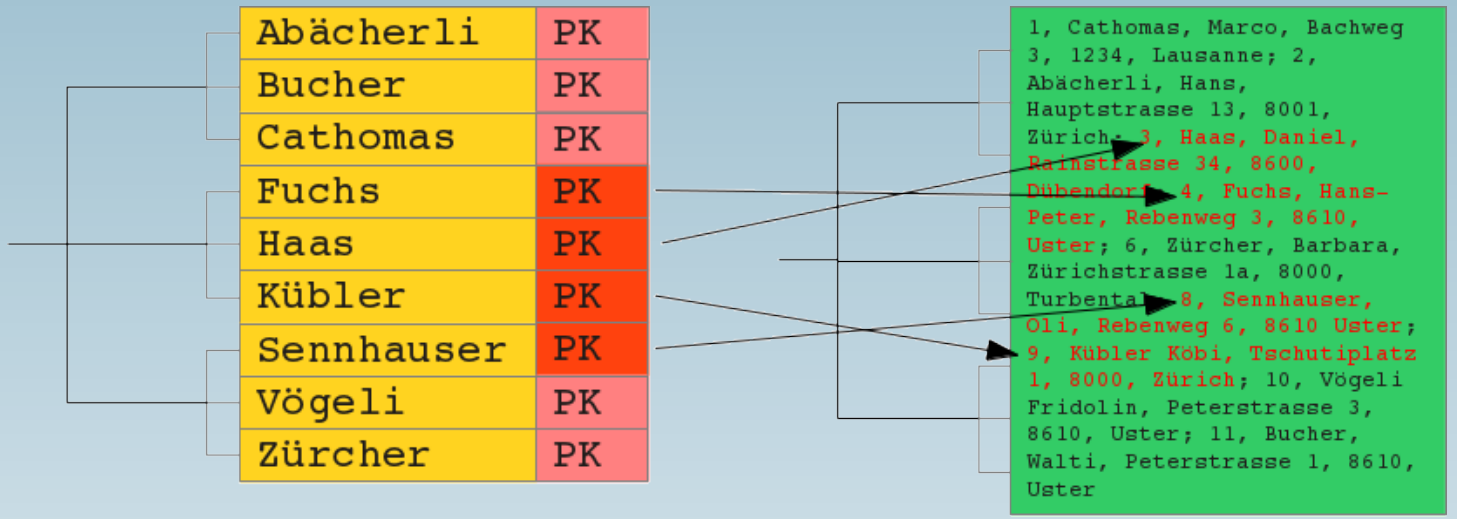
Indexieren

- Der Schlüssel zur besseren Query Performance sind Indices!
- Wo setzen wir Indices:
  - Jede Tabelle hat einen Primary Key!
  - Dort wo gejoined wird
  - Wo gute Filter vorhanden sind (WHERE a = ...)
- Spezialfälle
  - Covering Index
  - Index zu ORDER BY Optimierung
  - PK zur Verbesserung der Lokalität der Daten

Was sind gute Filter?

- Perfekter Filter: Primary Key, Unique Key
  - → 1 Treffer pro Wert
- Schlechter Filter: `ADD INDEX (gender)`
  - → 50/50 Verteilung
  - Kandidaten: Status, Gender, Solved, ...
- MySQL Optimizer KEINE Histogramme
  - → Optimizer liegt manchmal daneben
  - → Hints: `USE INDEX ()`, `IGNORE INDEX ()`

Warum ist falscher Index teuer?



- **Index Range Scan → random Access vs.**
- **Full table Scan → sequential Access**
- **Ab ca. 20% Daten wird Full Table Scan billiger**

ORDER BY Optimierung

- **Index kann für Sortierung verwendet werden**

```
EXPLAIN SELECT * FROM contacts AS c
 WHERE last_name = 'Sennhauser'
 ORDER BY last_name, first_name;
+-------+-----+-----------+------+-----------------------------------------------+
| table | type | key       | rows | Extra                                         |
+-------+-----+-----------+------+-----------------------------------------------+
| c     | ref  | last_name | 1561 | Using index condition; Using where; Using filesort |
+-------+-----+-----------+------+-----------------------------------------------+
```

Gewinn: 20 ms → 7 ms

```
ALTER TABLE contacts
  ADD INDEX (last_name, first_name);
+----------+-----+-------------+------+--------------------------+
| table    | type | key         | rows | Extra                    |
+----------+-----+-------------+------+--------------------------+
| contacts | ref  | last_name_2 | 1561 | Using where; Using index |
+----------+-----+-------------+------+--------------------------+
```

Covering Indexes

- **Index, der alle Spalten der Abfrage abdeckt:**

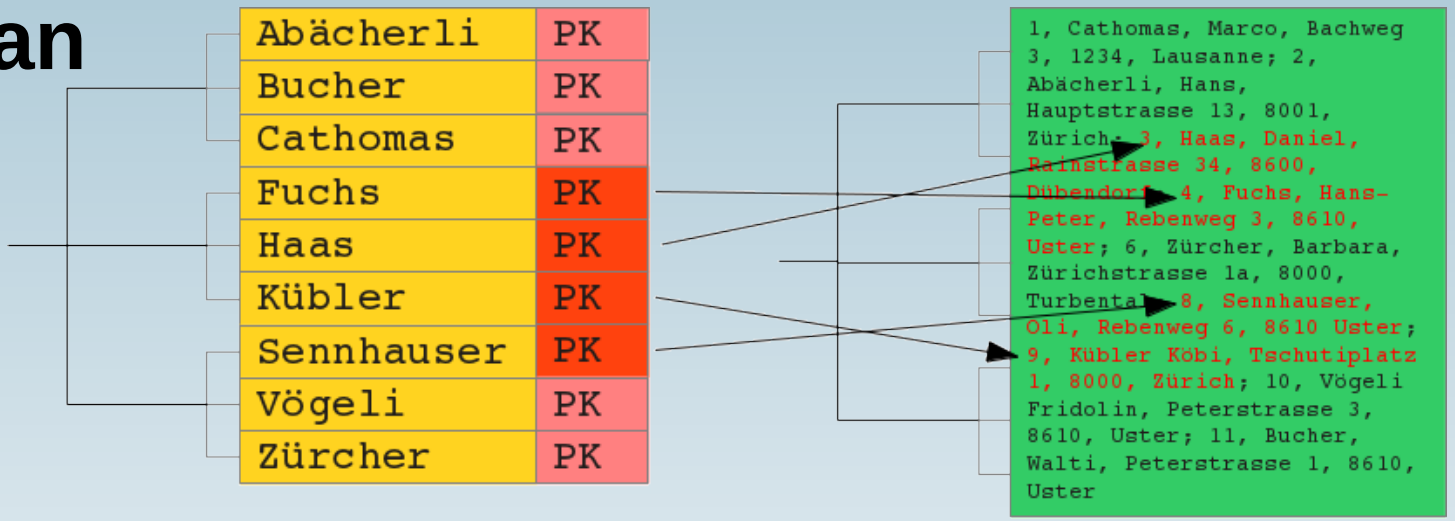
```
EXPLAIN
SELECT customer_id, amount
  FROM orders AS o
 WHERE customer_id = 59349;
+-------+------+-------------+------+-------+
| table | type | key         | rows | Extra |
+-------+------+-------------+------+-------+
| o     | re
+-------+---
```

Ja und?

```
ALTER TABLE orders
  ADD INDEX (customer_id, amount);
+-------+------+--------------+------+-------------+
| table | type | key          | rows | Extra       |
+-------+------+--------------+------+-------------+
| o     | ref  | customer_id_2 |   15 | Using index |
+-------+------+--------------+------+-------------+
```

Vorteil von Covering Indexes

- **Warum ist ein Covering Index so toll?**
- **Range Scan**



- **Full Index Scan:**

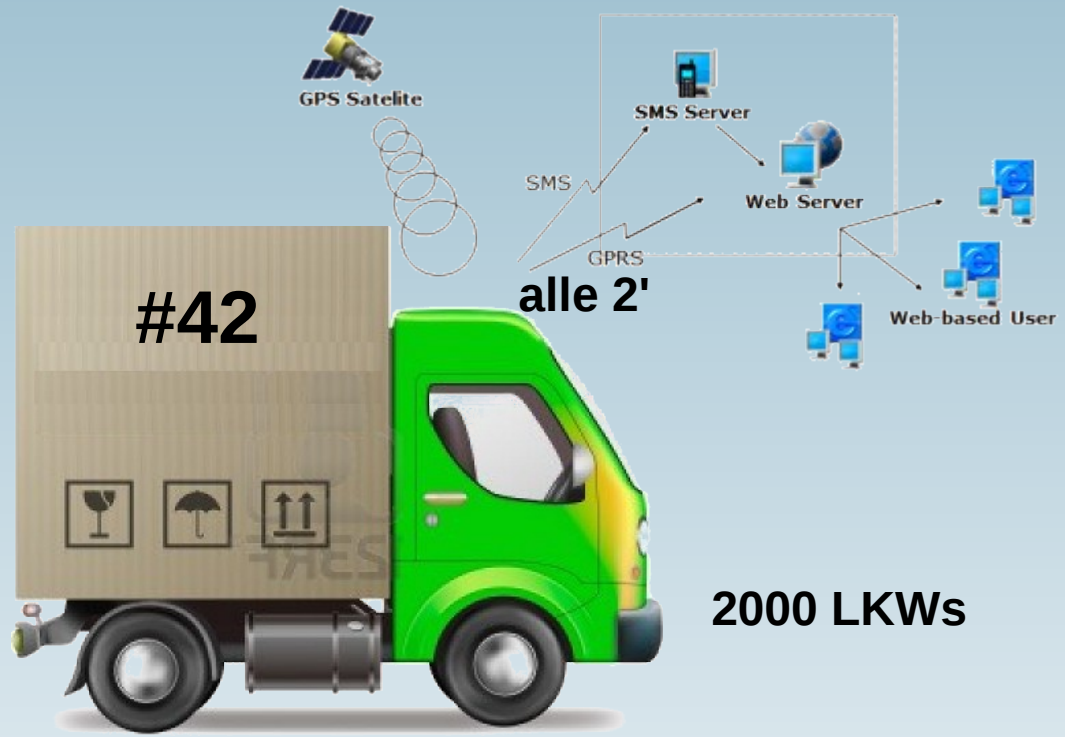
Lokalität der Daten

- Wie sind meine Daten physisch abgelegt?
- InnoDB: Index Clustered Table (IOT)
  - Row ist Teil des Primary Key
  - Rows sind sortiert wie Primary Key
- `AUTO_INCREMENT` ~= Sortierung nach Zeit!
- Oft gut:
  - Wenn heisse Daten = aktuelle Daten
- Schlecht für Zeitreihen:
  - Wenn heisse Daten = Daten pro Item über Zeit

Beispiel: InnoDB

www.fromdual.com

A_I	ts	v_id	xpos	ypos	...
1	17:30	#42	x,	y,	...
2	17:30	#43	x,	y,	...
3	17:30	#44	x,	y,	...
	...				
2001	17:32	#42	x,	y,	...
2002	17:32	#43	x,	y,	...
2003	17:32	#44	x,	y,	...



Q1: Δ in rows? ~ 2000 rows
A1: 1 row ~ 100 byte
Q2: Δ in bytes? ~ 200 kbyte
Q3: Default InnoDB block size? default: 16 kbyte
Q4: Avg. # of rows of car #42 in 1 InnoDB block? ~ 1
A2: 3 d and 720 pt/d → ~2000 pt ~ 2000 rec ~ 2000 blk
Q5: How long will this take and why (32 Mbyte)?
~ 2000 IOPS ~ 10s random read!!!
S: All in RAM or strong I/O system or ...?

über 3 Tage

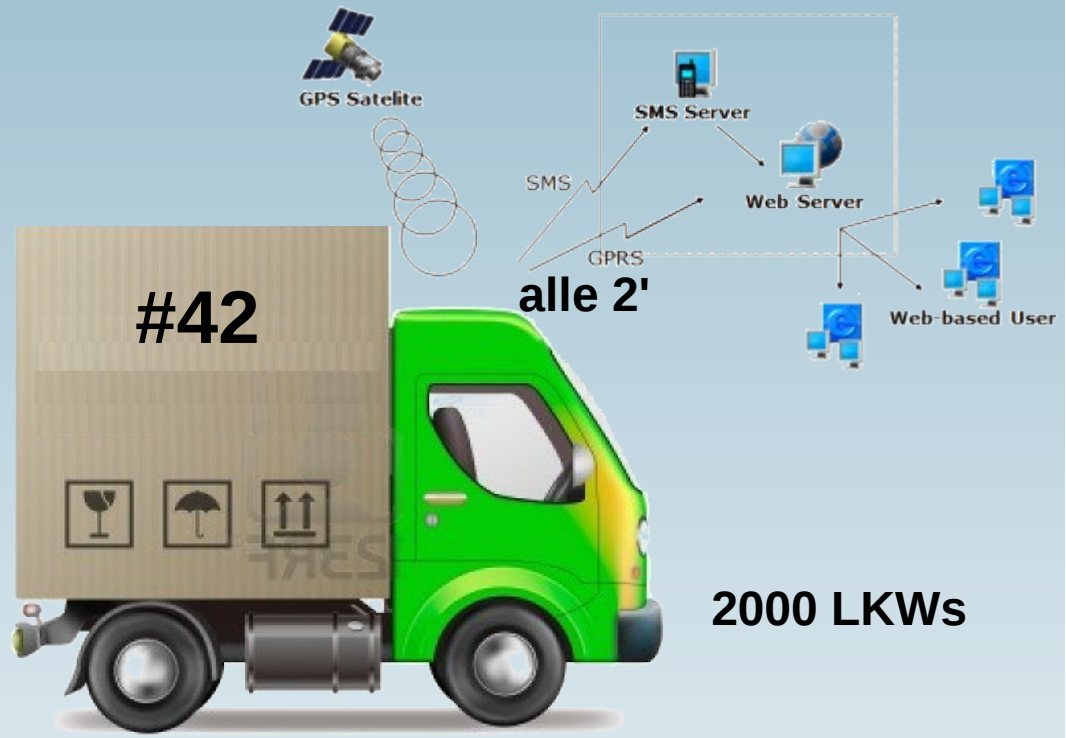
InnoDB PK rettet den Tag!

www.fromdual.com

```
ts v_id xpos ypos ...

17:30 #42 x, y, ...
17:32 #42 x, y, ...
17:34 #42 x, y, ...
          ...
17:30 #43 x, y, ...
17:32 #43 x, y, ...
17:34 #43 x, y, ...

          ...
17:30 #44 x, y, ...
```



Q1: Avg. # of rows of car #42 in 1 InnoDB block? ~ 120
A1: 3 d and 720 pt/d → ~2000 pt ~ 2000 rec ~ 20 blk
Q2: How long will this take and why (320 kbyte)?
~ 1-2 IOPS ~ 10-20 ms sequential read!
S: Wow f=50 faster! Any drawbacks?

über 3 Tage

Suche in Text


```
SELECT email_id FROM emails
WHERE email_body LIKE '%vertrag%';
```

- Was ist das Problem?

```
EXPLAIN SELECT email_id FROM emails WHERE email_body LIKE '%vertrag%';
+-------------+--------+------+---------------+------+---------+-------------+
| select_type | table  | type | possible_keys | key  | rows    | Extra       |
+-------------+--------+------+---------------+------+---------+-------------+
| SIMPLE      | emails | ALL  | NULL          | NULL | 1826340 | Using where |
+-------------+--------+------+---------------+------+---------+-------------+
1250 rows in 1050 ms
```

- MySQL kann Volltext Indexierung

Volltext-Indexierung

- Lösung: Volltext Index anlegen
- Ab 5.6 auch mit InnoDB möglich

```
ALTER TABLE emails ADD FULLTEXT INDEX (email_body);

EXPLAIN SELECT email_id FROM emails WHERE MATCH (email_body) AGAINST ('vertrag');
+-------------+--------+----------+---------------+------------+------+-------------+
| select_type | table  | type     | possible_keys | key        | rows | Extra       |
+-------------+--------+----------+---------------+------------+------+-------------+
| SIMPLE      | emails | fulltext | email_body    | email_body |    1 | Using where |
+-------------+--------+----------+---------------+------------+------+-------------+
1250 rows in 20 ms
```

- Architektur: Dokumente NICHT in DB
- Volltext Indexierung: Solr, Lucene, Elastic Search

Wir suchen noch:

www.fromdual.com

MySQL Datenbank Enthusiast/in für Support / remote-DBA / Beratung

Q & A

Fragen ?

Diskussion?

Wir haben Zeit für ein persönliches Gespräch...

- **FromDual bietet neutral und unabhängig:**
  - **Beratung**
  - **Remote-DBA**
  - **Support für MySQL, Galera, Percona Server und MariaDB**
  - **Schulung**

www.fromdual.com/presentations